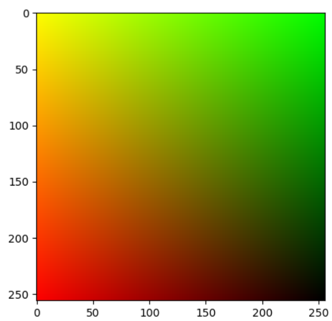
3DVC HW2

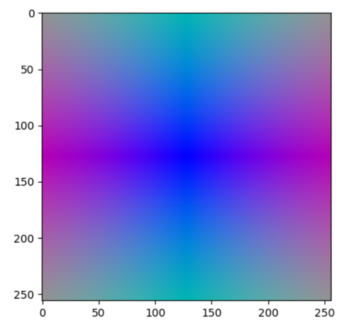
Problem1

Ray sampling

After implementing the function, we can get the result:



grid

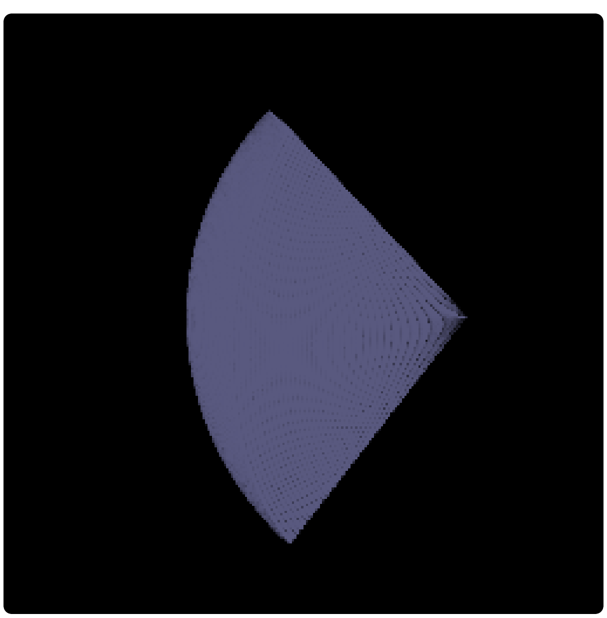


ray

The code is in the folder.

Point sampling

After implementing the functions, we get the result:

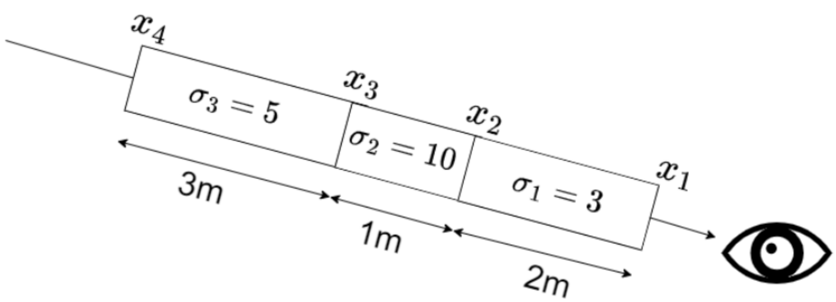


render_ray

The code is in the folder.

Theory of transmittance calculation

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$$

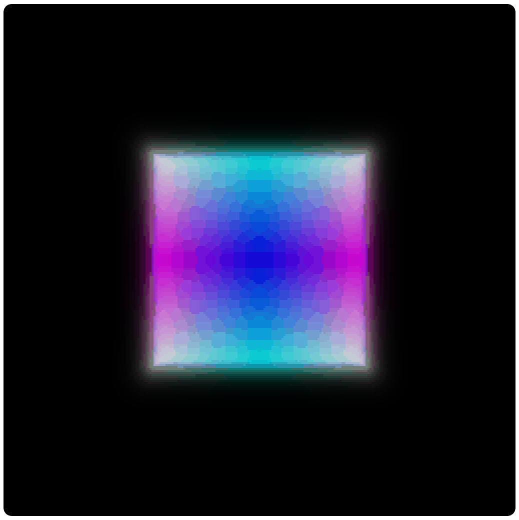
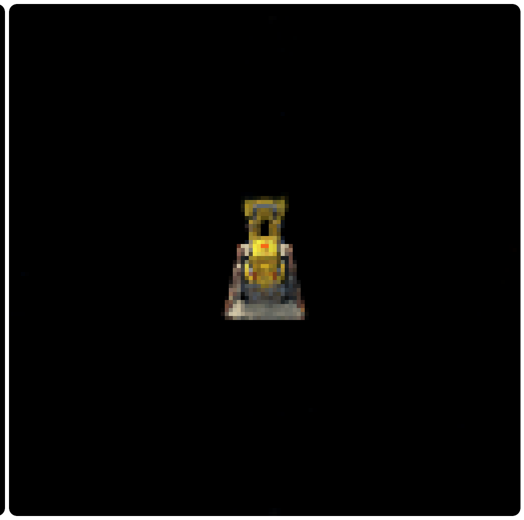


So we can know that:

$$T_1 = 1$$
$$T_2 = T_1 \times \exp(-\sigma_1 \delta_1) = e^{-6}$$
$$T_3 = T_2 \times \exp(-\sigma_2 \delta_2) = e^{-6} \times e^{-10} = e^{-16}$$
$$T_4 = T_3 \times \exp(-\sigma_3 \delta_3) = e^{-16} \times e^{-15} = e^{-31}$$

Rendering

After implementing the functions, we get the result:

The code is in the folder.

Problem2

3D losses

(a)

Proof.

We take note as :

$$A(\delta)=\{y\in\mathbb{R}:d(y,A)<\delta\}$$

1) When $h(X,Y)=0$, that means $X\subset Y(0)=Y$ and $Y\subset X(0)=X$, so $X=Y$.

2) $h(X,Y)=h(Y,X)$ is obvious.

3) We set $h(X,Z)=\inf\{\delta>0:X\subset Z(\delta);Z\subset X(\delta)\}=\delta$ and $h(X,Y)=\inf\{r>0:E\subset Y(r);Y\subset X(r)\}=r$.

It is easy to see that if $X\subset Y(\delta)$, we have $X(r)\subset Y(\delta+r)$.

So $Z \subset X(\delta), X \subset Y(r) \Rightarrow Z \subset X(\delta) \subset Y(\delta + r)$, $Y \subset X(r), X \subset Z(\delta) \Rightarrow Y \subset X(r) \subset Z(\delta + r)$. So we have that $h(Z, Y) \leq \delta + r$.

(b)

CDLoss Python 复制代码

```python
def forward(self, prediction, ground_truth):
        # TODO: Implement CD Loss
        # Example:
        #     cd_loss = torch.tensor(0, dtype=torch.float32, device=prediction.device)
        #     return cd_loss

        cd_loss = torch.sum(torch.norm(prediction - ground_truth, p=2))
        return cd_loss
```

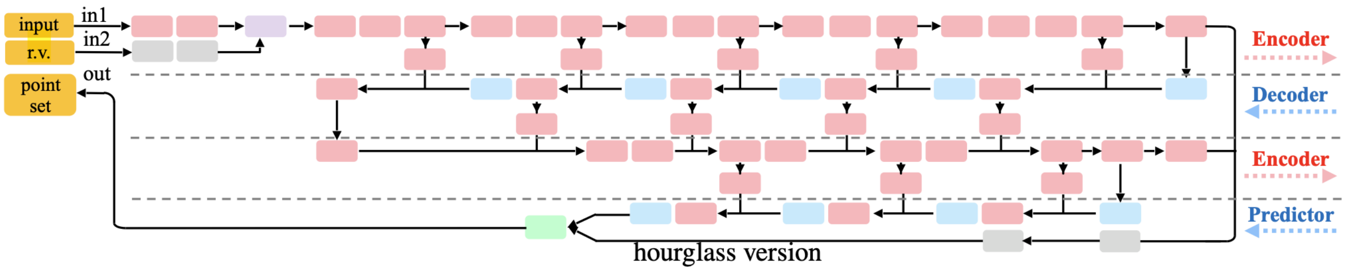
HDLoss Python 复制代码

```python
def forward(self, prediction, ground_truth):
        # TODO: Implement HD Loss
        # Example:
        #     hd_loss = torch.tensor(0, dtype=torch.float32, device=prediction.device)
        #     return hd_loss
        # Compute the Euclidean distance map
        dist_map = torch.cdist(prediction, ground_truth, p=2)

        hd_dist = torch.max(torch.min(dist_map, dim=1)[0])
        hd_loss = hd_dist

        return hd_loss
```

Network design

(a)

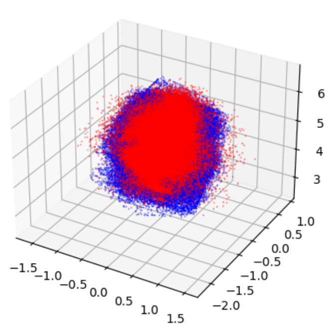
I follow the PointOutNet structure. The code can be seen in the model.py.



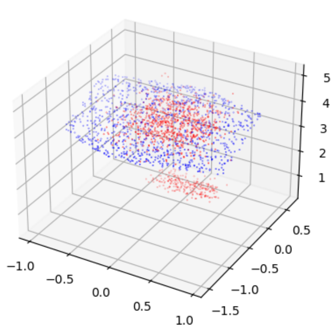
(b,c)

By using the same epoch(50), learning rate(0.01) and batch size(64), I get these result:

	CDLoss	HDLoss
train loss clean	190.8165	5.9669
test loss clean	121.3022	7.9568
train loss noisy		
test loss noisy		



CD_Train



CD_test

(d)

From the expriement we can find that the HD loss is more sensitivity to the outliers. When the data have some noise, the performance of HD loss model is quite bad.

And we find that the HD loss is not differentiabe. And it lack of the interpretation.

Problem 3

MLS constraints

MLS interpolation

Marching Cube